GUI

Vi började vårt projekt genom att skapa en mall som mer eller mindre var en kopia av det projektet som Jakob visade på lektionen. Vilket vi snabbt insåg var lite tråkigt. Det var då det största designbeslutet i GUI gjordes. Vi ville ha en snygg skalande layout. Valet landande på att göra en FlowLayoutPanel. I den skulle alla produkterna läggas till. Sen skapade vi varje produkt som en TableLayoutPanel med tillhörande labels, bild och knapp. Istället för att bara göra kvittot i en MessageBox ville vi ha mer möjlighet att designa och därför gjorde vi även ett nytt formulär.

Klasstrukturer

Vi började projektet med en klass för våra produkter som vi sedan sparade i en lista. Varje gång en produkt lades till i vår varukorg sparades objektet med ett heltalsvärde som kvantitet i en Dictionary. Vi insåg ganska snabbt att vi ville abstrahera ner det i ännu flera klasser med en egen klass för våra Cart-objekt som vi istället på samma sätt som produkterna sparar i en lista. Detta för att få en mer enhetlig struktur igenom hela programmet mellan alla våra klasser däribland våran DiscountCode klass som vi valde att designa på liknande sätt.

Dela värden mellan klasser

Vi har valt att använda oss av statiska variabler vid uppstrukturerandet av vårat program, detta för att enkelt få tillgång till värden mellan dom olika klasserna som vi har. Vi insåg dock ju längre vi kom och när förståelsen för hur dessa statiska variablerna ökade, att vi i en framtida version av programmet istället skulle göra om det till ett mer avgränsat upplägg med hjälp av konstruktorer för att skicka värden mellan de olika klasserna. Vilket även hade minskat förekomsten av spaghettikod.

Vi stötte till exempel på ett problem med dom statiska variablerna när vi skapade upp CheckoutWindow formuläret. Vi ville ha möjligheten att stänga ner detta fönster för att sedan få upp det igen under samma körning. Det visade sig då att med de statiska variablerna vi hade på knappens event som aktiverar rabattkoden. De eventen "levde" kvar under nedstängningen och öppnandet av formuläret. Då kördes klickeventet så många gånger som formuläret hade skapats.

Som vi förstod det. Om vi skulle använda detta programmet på ett storskaligt plan så hade det varit tvunget att vara icke statiska variabler. Detta då inga ändringar som en användare gör ska kunna påverka andra användare.

Arv

När vi började med vår design så började vi fundera över hur man vet vilken produkt som är vilken i listan. Vi började lägga in varje produktobjekt i Tag-egenskapen i Button. Men detta märkte vi, även kunde åstadkommas med arv. Vi skapade en ny klass som ärvde av Button. Där lade vi även till vår egna egenskap, Product. Då kunde vi enkelt bara lägga till ett värde i den egenskapen då vi skapade upp produktutbudet.

Efter att ha fått mer övergripande förståelse för vårat program. Tänker vi att vid en omstrukturering skulle våra produktklasser ärva från en övergripande produktklass. Detta för att sen lätt kunna lägga till ett sortiment av olika produkter där dessa då ärver från en ursprunglig klass för att sedan kunna ha några egna specifika egenskaper.